

Praxisbeispiel MiFare

in der Vorlesung Systemsicherheit

Johannes Schlumberger
asso@0xbadc0ffee.de

Friedrich-Alexander-Universität Erlangen/Nürnberg

11. Dezember 2008



Inhalt

- 1 Überblick über MiFare
- 2 Anwendungsszenarien
 - Eintrittskarten
 - Warentracking
 - Fahrkartensysteme
 - Zugangssysteme
 - Bezahlungssysteme
 - Weiteres
- 3 Angriffsvektoren
 - Emulatoren
 - Cryptoanalytische Angriffe
- 4 Ausblick



Überblick über MiFare



technische Grundlagen

- passive, kontaktlose Smartcard
- RFID-Standard (Radio Frequency Identification)
- ISO/IEC 14443-2:2001(E) Standard
 - 1 physikalische Eigenschaften (Größe, Leistungsaufnahme, etc)
 - 2 Codierung der Funkübertragung
 - 3 Protokoll (Antikollision, etc) und Kommandosatz
 - 4 Frequenzbereich (13.56 Mhz)
- Karte hat nichtflüchtigen Speicher
- modellabhängig gegenseitige Authentifizierung vor Speicherzugriffen
- Crypto-1, shared secret (vom Betreiber des Systems festgelegt)
- eindeutige UID ab Werk fest



Kartenarten

- MiFare ist ein Warenzeichen von NXP-Semiconductors (Philips-spin-off)
- weltweit Milliarden Karten im Einsatz



Kartenarten

- MiFare ist ein Warenzeichen von NXP-Semiconductors (Philips-spin-off)
- weltweit Milliarden Karten im Einsatz

	Speicher	Crypto	Sonstiges	UID
Ultralight	64 Byte	-	32 OTP-Bits	7 Byte
Classic (Standard)	4 KByte	Crypto-1	-	4 byte

- Weitere Varianten z.B. MiFare-DESFire - später



Anwendungsszenarien



Eintrittskarten

Fussballweltmeisterschaft 2006

- RFID-Karten
- personalisiert über Kundennummer und UID
- nur optisch lesbar auch Name des Inhabers
- unerlaubte Weitergabe invalidiert das Ticket
- RFID-Lesegeräte an den Stadioneingängen
- Schwarzmarkthandel eingeschränkt



Warentracking

- in Kaufhäusern, Warenlagern, bei grossen Abnehmern, etc
- Produkt/Paket hat aufgedruckten RFID-Chip
- einfache Erfassung bei Lieferung/Kauf (WALMART)
- eventuell automatisierte Abrechnung
- eigentlich nur UID nötig



Warentracking

- in Kaufhäusern, Warenlagern, bei grossen Abnehmern, etc
- Produkt/Paket hat aufgedruckten RFID-Chip
- einfache Erfassung bei Lieferung/Kauf (WALMART)
- eventuell automatisierte Abrechnung
- eigentlich nur UID nötig

Privacy

Chips werden nicht deaktiviert.

Tracking von Kunden auch außerhalb des Geschäfts durch Zusammenführen von Leserdaten möglich.

Grundsätzliches Problem der RFID-Technologie.



Fahrkartensysteme

- MiFare Ultralight
 - für einfache Fahrten (alle gleichartig, keine Tarifzonen)
 - bis zu 32 Einzelfahrten einfach per OTP realisierbar
 - nicht gebuchte Fahrten bei Kauf entwerten



Fahrkartensysteme

- MiFare Ultralight
 - für einfache Fahrten (alle gleichartig, keine Tarifzonen)
 - bis zu 32 Einzelfahrten einfach per OTP realisierbar
 - nicht gebuchte Fahrten bei Kauf entwerten
- MiFare Classic
 - für komplexere Fahrkarten
 - Karte hält Datum, Zugbindung, Personenbindung evtl. Preis
 - verschlüsselt abgelegt
 - Kontrolleuer mit Lesegerät ausgestattet
- Authentizität der Karten anhand der UID-Range prüfbar



Zugangssysteme

- verschiedene Variationen denkbar
- Leser speichern UID
- Leser speichern UID und Berechtigungen
- Berechtigung verschlüsselt auf der Karte abgelegt
- Backendsystem prüft anhand der UID die Berechtigungen
- Backendsystem prüft anhand der UID und verschlüsselter Kartendaten



Bezahlsysteme

in vielen Mensen, ...

- MiFare Classic Karten
- best. Kommandos zum sicheren Lesen und Schreiben von Geldbeträgen
- “sicher” bzgl. Datenintegrität und Vertraulichkeit
- Lesegeräte modifizieren die Geldbeträge
- beim Aufladen und Bezahlen
- verschiedene Schlüssel für Schreiben und Lesen möglich



Weitere Anwendungsmöglichkeiten

Es existieren zahlreiche weitere Anwendungen

- Autoschlüssel
- sog. Firmenkreditkarten
- Personalausweise/Reisepässe (EU, UNO Empfehlung)
- Bücherei/Videotheksausweise
- 24h Videotheken
- Kopierkarten, Getränkeautomaten, ...
- UMTS (Universal Mate Tracking System)
- ...



Angriffsvektoren



Kartenemulator

Wenn man einen Emulator hätte

- UID einstellbar
- Inhalte frei wählbar
- Karten clonbar, wenn Inhalt des Originals lesbar
- immer möglich für Ultralight, Crypto-1 bei Classic



Kartenemulator

Wenn man einen Emulator hätte

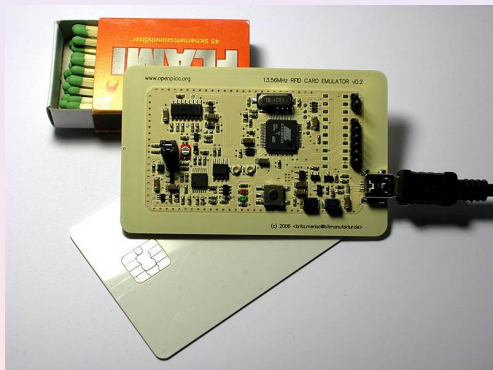
- UID einstellbar
- Inhalte frei wählbar
- Karten clonbar, wenn Inhalt des Originals lesbar
- immer möglich für Ultralight, Crypto-1 bei Classic

Tada!

Solche Projekte existieren
und sogar unter freien Lizenzen (SW und HW-Design)
(z.B. <http://openpicc.org>, <http://proxmark.org>)



OpenPICC



- MiFareClassic Chip
- Atmega
Microcontroller
- 32-bit-ARM
- 64 Kbyte SDRAM
- 256 Kbyte Flash
- kontrolliert MiFare
Chip
- voll programmierbar



Lesegerät OpenPCD



- Schwesterprojekt des OpenPICC
- vollständige Kontrolle über RFID-Signal
- weitere Hardware on board



Cryptoanalytische Angriffe

Wenn man den Cipher gebrochen hätte

- freie Manipulation von Inhalten auch auf MiFare Classic
- Erzeugung gültiger verschlüsselter Inhalte auf dem Emulator
- beliebige verschlüsselte Inhalte mit beliebiger UID
- Auslesen fremder Kartendaten

⇒ Cipher zu brechen ist ein lohnenswertes Ziel



Crypto-1

- proprietärer Cipher von NXP Semiconductors
- Website verrät:
 - stromsparender als AES
 - “approved authentication”
 - “advanced security levels”
 - 48-Bit key (Brute Force: $2^{48} \text{keys} \frac{25 \text{ms}}{\text{key}} \approx 230.000 \text{ Jahre}$)
- wenig Information
- security by obscurity



Crypto-1

- proprietärer Cipher von NXP Semiconductors
- Website verrät:
 - stromsparender als AES
 - “approved authentication”
 - “advanced security levels”
 - 48-Bit key (Brute Force: $2^{48} \text{keys} \frac{25 \text{ms}}{\text{key}} \approx 230.000 \text{ Jahre}$)
- wenig Information
- security by obscurity

Kerckhoffs Maxime

Die Sicherheit eines Verschlüsselungsverfahrens beruht auf der Geheimhaltung des Schlüssel, und nicht auf der Geheimhaltung des Verschlüsselungsalgorithmus.



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen
- 3 Fotos machen



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen
- 3 Fotos machen
- 4 Bildanalysesoftware schreiben die best. Gatemuster sucht



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen
- 3 Fotos machen
- 4 Bildanalysesoftware schreiben die best. Gatemuster sucht
- 5 Flipflopzeilen / XOR-Dichte hoch? / autonomer Teil?



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen
- 3 Fotos machen
- 4 Bildanalysesoftware schreiben die best. Gatemuster sucht
- 5 Flipflopzeilen / XOR-Dichte hoch? / autonomer Teil?
- 6 Schaltung reproduzieren



Reverse Engineering

Gruppe CCC-naher Hacker

- 1 MiFarekarte in Aceton auflösen \Rightarrow Chip
- 2 mittels Poliergerät für Optiklinsen abschleifen
- 3 Fotos machen
- 4 Bildanalysesoftware schreiben die best. Gatemuster sucht
- 5 Flipflopzeilen / XOR-Dichte hoch? / autonomer Teil?
- 6 Schaltung reproduzieren

\Rightarrow Schaltplan des Algorithmus



Was fällt auf?

1 PRNG

- nur 16 bit (Gleicher Zustand alle ~ 0.6 Sekunden)
- Wert hängt von Anschaltezeit ab
- \Rightarrow Timing durch Open{PICC,PCD} kontrollierbar
- \Rightarrow Zufallszahlen kontrollierbar



Was fällt auf?

1 PRNG

- nur 16 bit (Gleicher Zustand alle ~ 0.6 Sekunden)
- Wert hängt von Anschaltezeit ab
- \Rightarrow Timing durch Open{PICC,PCD} kontrollierbar
- \Rightarrow Zufallszahlen kontrollierbar

2 LFSR (linear feedback shift register) basierte Crypto

- rückgekoppeltes Schieberegister
- jeden Takt wird rausfallendes Bit mit anderen verxort und rückgekoppelt
- mathematisch leicht beschreibbar (rein linear)
- üblicherweise mit nichtlinearen Funktionen gekoppelt (nicht hier)
- Charakteristisches Polynom des PNRG-LFSR ist Beispiel von Wikipedia



Was heisst das?

- sehr schwacher PRNG
- keine komplexen (nichtlinearen) Funktionen in der Schaltung
- mathematisch gut beschreib- und handhabbar
- insbes. möglich den Zustand des LFSR voraus und rückwärts zu berechnen, wenn Zustand zu einem Zeitpunkt bekannt ist



Was heisst das?

- sehr schwacher PRNG
- keine komplexen (nichtlinearen) Funktionen in der Schaltung
- mathematisch gut beschreib- und handhabbar
- insbes. möglich den Zustand des LFSR voraus und rückwärts zu berechnen, wenn Zustand zu einem Zeitpunkt bekannt ist

⇒ known-plaintext-attack auf den key vermutlich deutlich effizienter als brute-force



FPGA-Cluster



- einfach auf FPGAs zu implementieren
- \$100-Cracker
- 14 FPGA-Boards
- ~1 Woche
- aber: known plaintext nötig



Algebraischer Angriff

Summary of Claims

We can recover the full 48-bit key of MiFare Crypto-1 algorithm in 200 seconds on one 1.66 GHz Centrino CPU, given 1 known IV and 50 output bits (from one single encryption).

With 4 chosen IVs we can recover the key in 12 seconds.

(Karsten Nohl e.a., April 2008)



Algebraischer Angriff

Summary of Claims

We can recover the full 48-bit key of MiFare Crypto-1 algorithm in 200 seconds on one 1.66 GHz Centrino CPU, given 1 known IV and 50 output bits (from one single encryption).

With 4 chosen IVs we can recover the key in 12 seconds.

(Karsten Nohl e.a., April 2008)

- Algorithmus in einfache Schritte zerlegen
- symbolische boolesche Gleichungen über \mathbb{F}_2^x
- Einschluss von IV, keybits, outputbits
- Übersetzung in SAT-Problem
- SAT-Solver



LFSR-Rollback

- Crypto-1 basiert ebenfalls auf LFSR ohne nichtlineare Funktionen
- shared secret ist IV
- Rückrechnung möglich wenn Zustand zu einem Zeitpunkt bekannt
- Sniffing in der Authentifizierungsphase
- daraus aktuellen Zustand berechnen
- anschliessend rollback (in Software)
- \Rightarrow Schlüssel
- Kommunikation vollständig entschlüsselbar (eavesdropping)



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt “Request card”
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID
- 6 Karte antwortet mit Typ (MiFare Classic)



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID
- 6 Karte antwortet mit Typ (MiFare Classic)
- 7 Reader schickt auth-request



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID
- 6 Karte antwortet mit Typ (MiFare Classic)
- 7 Reader schickt auth-request
- 8 Karte antwortet mit challenge



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID
- 6 Karte antwortet mit Typ (MiFare Classic)
- 7 Reader schickt auth-request
- 8 Karte antwortet mit challenge
- 9 Reader antwortet mit challenge und response



Authentifizierungsprotokoll

gegenseitige Authentifizierung

- 1 Reader loopt "Request card"
- 2 Karte bekommt Strom und meldet sich
- 3 Reader fragt ab welche Karten im Feld sind
- 4 Karten antworten mit UID
- 5 Reader selektiert eine Karte über UID
- 6 Karte antwortet mit Typ (MiFare Classic)
- 7 Reader schickt auth-request
- 8 Karte antwortet mit challenge
- 9 Reader antwortet mit challenge und response
- 10 Karte schickt response an Leser



im Trace

Step	Sender	base ₁₆	Bedeutung
01	Leser	26	req type A
02	Karte	04 00	answer req
03	Leser	93 20	select
04	Karte	c2 a8 2d f4 b3	uid, checksumme
05	Leser	93 70 c2 a8 2d f4 b3 ba a3	select(uid)
06	Karte	08 b6 dd	MiFare 1K
07	Leser	60 30 76 4a	auth(block 30)
08	Karte	42 97 c0 a4	c_K
09	Leser	7d db 9b 83 67 eb 5d 83	$c_R \oplus ks_1, r_R \oplus ks_2$
10	Karte	8b d4 10 08	$r_K \oplus ks_3$



Weitere Erkenntnisse

Bitflipping (UID, c_K), wiederholtes sniffing und Analyse

- Crypto-1 initialisiert während Authentifizierungsphase
- Key in LFSR (7)
- dann $c_K \oplus uid$ in LFSR shiften (9)
- c_R in LFSR shiften (10)
- Möglichkeit jetzigen LFSR-Zustand zu bestimmen existiert (11)



Code

```
uint64_t recover_key(struct sniffdata *sd)
{
    struct Crypto1State revstate;
    uint32_t ks2, ks3;
    uint64_t lfsr;

    /*ks2 = suc2(tag_challenge) ^ encAr (suc 2 = 2*32 steps)*/
    ks2 = sd->reader_response_enc ^ prng_successor(sd->tag_challenge, 64);
    /*ks3 = suc3(tag_challenge) ^ encAt (suc 3 = 3 * 32 steps)*/
    ks3 = sd->tag_response_enc ^ prng_successor(sd->tag_challenge, 96);

    /*reverse, and compute the current lfsr state from keystream*/
    lfsr_recovery(&revstate, ks2, ks3);

    /*rollback lfsr to get key*/
    lfsr_rollback(&revstate, 0, 0);
    lfsr_rollback(&revstate, 0, 0);
    lfsr_rollback(&revstate, sd->reader_challenge_enc, 1);
    lfsr_rollback(&revstate, sd->uid ^ sd->tag_challenge, 0);

    crypto1_get_lfsr(&revstate, &lfsr);
    return lfsr;
}
```



Emulator und Crypto-1

Ursprüngliche Sicherheitsziele

- Datenvertraulichkeit
- Datenintegrität
- Authorisierung
- Authentifizierung
- Verbindlichkeit



Emulator und Crypto-1

Ursprüngliche Sicherheitsziele

- Datenvertraulichkeit
- Datenintegrität
- Autorisierung
- Authentifizierung
- Verbindlichkeit

KTHNXBYE

Alle Sicherheitsziele verletzt und angreifbar!
Das System ist restlos offen.



Ausblick



Lösungen?

- MiFare DESFire
 - können 3DES/AES
 - neue Leser die Classic/DESFire können
 - aber Austausch aller Reader (und aller Karten) notwendig
 - das kostet!



Lösungen?

- MiFare DESFire
 - können 3DES/AES
 - neue Leser die Classic/DESFire können
 - aber Austausch aller Reader (und aller Karten) notwendig
 - das kostet!
- Backendsysteme
 - alle Transaktionen auf Schattenkonten
 - laufende Prüfung durch Backendsystem
 - Alarm bei Ungereimtheiten (Fehlalarm)
 - Daten auf Karte unsinnig
 - Leser müssen dauerhaft online sein
 - das kostet!



Lektion gelernt?

- Kerckhoffs Maxime
- security by obscurity hilft nur kurzzeitig
- fehlende “peer reviews” kosten später richtig Geld
- MiFare Classic nur für Kleinstwerte ausreichend sicher



Lektion gelernt?




- Kerckhoffs Maxime
- security by obscurity hilft nur kurzzeitig
- fehlende “peer reviews” kosten später richtig Geld
- MiFare Classic nur für Kleinstwerte ausreichend sicher

... und im Alltag

Augen offenhalten nach interessanten RFID Anwendungen ;-)



Literatur

-  Henryk Plötz, *MiFare Classic - eine Analyse der Implementierung*. Diplomarbeit, Humboldt-Universität zu Berlin, 2008.
-  Flavio D. Garcia e.a., *Dismantling MiFare Classic*. Radboud University Nijmegen, 2008.
-  Karsten Nohl, Henryk Plötz, *MiFare, little security, despite obscurity*. Präsentation auf der 24C3 des Chaos Computer Clubs in Berlin, 2007.
-  Auguste Kerckhoffs, *La cryptographie militaire*. Journal des Sciences Militaires IX, 5-38, 1883.
-  Karsten Nohl, Henryk Plötz, D. Evans, starbug *Reverse-engineering a cryptographic RFID tag*. USENIX Security 2008, 2008.

